IN THE SPECIFICATION

Please amend the specification as follows:

The two paragraphs beginning at page 11, line 19 are amended as follows:

One embodiment of a high availability (HA) computing system 10 is shown in Fig. 1. System 10 includes two or more computing ~~systems~~ servers 12 connected over a network 14 to clients 16. In normal operation, ~~both systems~~ servers 12 in a cluster can be active, working as if they were independent servers. In the event of a failure, however, the surviving ~~system~~ server 12 takes over the services of the failed system, transparently fulfilling requests from clients 16 on network 14. In one embodiment, system 10 supports heterogeneous clusters of servers 12, preserving the investment in existing computing infrastructure.

In the embodiment shown in Fig. 1, the two ~~systems~~ servers 12 are connected to both a public network 14 and a private network 18. Clients 16 use public network 14 to access services from the cluster. Software running on each server 12 use private network 18 to exchange heartbeat and other control messages. In one embodiment, private network [[12]] 18 comprises a serial communication network with a serial multiplexor 20 interconnecting the ~~nodes~~ servers 12 to the private network 18. In the event of a server or application failure, the surviving ~~system~~ server 12, if appropriately configured, assumes the public network address of the failed ~~system~~ server 12 and answer requests from clients 16 on network 14. In one embodiment, clients 16 perceive the failover process as a rapid reboot of the failed primary server.

The paragraph beginning at page 12, line 31 is amended as follows:

Instances of CMS [[12]] 32 communicate with one another using Cluster Membership Protocol 36. An instance of a CMS service is also referred to as a Cluster Management Daemon (CMD). As far as the Cluster Membership Service is concerned, nodes are represented by the CMD processes that run on them. The failure of such a CMD is interpreted as the failure of the node. As an example of one aspect of the operation of a CMD, in order to be able to reach agreement the servers may chose to follow a two-phase commit protocol. In that case the two-phase commit protocol will be a component of the Cluster Membership Protocol 36. Membership Protocols usually contain three phases:

The paragraph beginning at page 23, line 2 is amended as follows:

GCD monitors the *existence* and *liveliness* of all processes within a group. Group process failures trigger a group membership change, with the failed process being reported as exiting in an *unknown* state. Note that the specification uses the terms *group member* and *application instance* interchangeably since they refer to the same entity - an application process registered with GCS 34 at a ~~node~~ server 12.

The paragraph beginning at page 24, line 30 is amended as follows:

When a critical application instance exits without unregistering or fails to respond to monitoring in a timely fashion, the local GCD 34 may exit, causing the local CMS 32 instance to exit, which in turn causes the node the instances were running on to be physically reset. Exit or failure of a non-critical application instance is treated as an implicit unregister. The remaining GCDs on other ~~nodes~~ servers 12 are notified of the change in the group's membership.

The paragraph beginning at page 26, line 1 is amended as follows:

The Delta protocol can be summarized as follows. An initiator GCD node sends a message to the GCD coordinator node. The coordinator sends this message to the other GCD nodes in the cluster and waits for an acknowledgment from these nodes. This is the first phase *or proposal* phase. After receiving an acknowledgment, the coordinator sends out a *commit* message to the other GCDs 34 on ~~nodes~~ servers 12 and then waits for acknowledgment of this message. The commit phase is the second phase. All messages within a specific group are serialized through the coordinator, allowing a global ordering of messages, within a group. Unlike the traditional two-phase commit algorithm, an arbitrary node may not abort either a proposal or a commit.

The paragraph beginning at page 27, line 1 is amended as follows:

b. Initiator failure: If the message is in the coordinator's serializing proposal queue, the coordinator will not send out the message. The coordinator will filter out all the messages in the pending message queue from the initiator. If the message proposal

has already been sent, the message will be delivered to all remaining GCD 34 nodes. Only after all ~~nodes~~ servers 12 have seen this message will new memberships (including the membership containing the failed initiator) be delivered.

The paragraph beginning at page 27, line 16 is amended as follows:

2. Task 2: The coordinator sends a proposal for the message (ostensibly the message) to the remaining GCD server nodes. The message is passed from one server node to the next. The message proposal loops back to coordinator after being passed on by all the nodes. Each of the nodes puts the message proposal in its proposal buffer. There is a proposal time-out for each message. If the time-out value is reached, the coordinator resends the proposal message. ~~Nodes~~ Servers 12 who have already seen the proposal message continue to pass it along, as many times as necessary. These failures are possible and considered:

The paragraph beginning at page 30, line 30 is amended as follows:

- The protocol assumes that if the message reaches a particular GCD 34, it will be sent to the clients of that ~~node~~ server 12.

The paragraph beginning at page 34, line 1 is amended as follows:

In some embodiments, GCS 34 needs to obtain information about node membership of the cluster from the CMS [[12]] 32 layer. To achieve this, a GCD 34 process needs to register with CMS [[12]] 32 at startup and then periodically ask CMS for node membership changes. After registering with CMS, GCD 34 sets up several ways of receiving information from CMS periodically.

The paragraph beginning at page 34, line 27 is amended as follows:

CMS [[12]] 32 can deliver 3 types of membership notifications to GCD 34. They are:

The paragraph beginning at page 36, line 22 is amended as follows:

In one embodiment, clustered servers 12 share storage either on RAID or mirrored disks. One such embodiment is shown in Fig. 2, where a RAID system 20 is connected to each of the cluster ~~nodes~~ servers 12 over a SCSI channel 22. In addition, a backup SCSI channel 24 is provided for fail over of channel 22. A shared storage subsystem allows either server 12 to assume control of the data in the event of a failure. The software is designed so that one machine will automatically take over the other system's filesystems in the event of a failure.

The six paragraphs beginning at page 36, line 33 and continuing to page 38, line 3 is amended as follows:

In one embodiment, system 10 can be configured either as active/standby or dual active. In an active/standby configuration, one ~~machine~~ server 12 runs the workload while the other ~~machine~~ server 12 is in standby mode, prepared to take over if the primary server fails. In the dual active configuration, both servers 12 are doing useful work while acting as each other's backup. In the event of a failure, the logical backup server is doubly loaded, running not only its own workload, but also that of the failed server.

In some embodiments, part of the HA software is the application monitor. In one embodiment, the application monitor is a daemon process that monitors heartbeat messages and executes application-specific scripts for monitoring applications. An instance of this daemon runs on each ~~node~~ server 12 of the cluster. In the event of a server or application failure, the daemon on the surviving ~~system~~ server 12 executes software to cause the surviving ~~system~~ server to assume the public network address of the failed ~~system~~ server 12 and answer requests from clients 16 on network 14. In one embodiment, as noted above, clients 16 perceive the failover process as a rapid reboot of the failed primary server.

Yet another embodiment of system 10 is shown in Fig. 3. In Fig. 3, up-to eight servers are connected to both a public network 14 and a private network 18. Clients 16 use public network 14 to access services from clustered system 10. The HA software uses private network 18 to exchange heartbeat and other control messages. In the event of a server or application failure, one of the surviving ~~systems~~ server 12 assumes the public network address of the failed system and responds to the client requests on network 14. Clients 16 perceive the failover process as a rapid reboot of the system to which they were connected.

In one embodiment, the HA software is built on top of a Cluster Administration and Membership Services (CAMS) layer. The CAMS layer is highly sophisticated distributed

software that efficiently controls the applications in a cluster. This layer enables efficient addition and deletion of systems and applications from a cluster and also intelligently recovers the cluster from network partitioning.

In one embodiment, the clustered ~~nodes~~ servers 12 share storage either on RAID 20 or mirrored disks. A shared storage subsystem allows multiple servers 12 to assume control of the data in the event of a failure when the filesystems are automatically made available on the system(s) where their corresponding applications are resumed.

In one embodiment, a system 10 cluster can be configured either in N x 1 or in N x N mode. In an N x 1 configuration (such as is shown in Fig. 3), N ~~machines~~ servers 12 run various mission critical applications while one machine is in standby mode, prepared to take over if any of the primary N servers fail. This configuration ensures that your environment sees no performance degradation even after the failure of a server 12.